CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1752

Multi-organization scheduling approximation algorithms

Johanne Cohen[1], Daniel Cordeiro[2], Denis Trystram[3,4,*,†] and Frédéric Wagner[3]

[1]*Laboratoire d'Informatique PRiSM, Université de Versailles St-Quentin-en-Yvelines, 45 Avenue des États-Unis, 78035 Versailles Cedex, France*
[2]*LIG, Grenoble University, 51 Avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France*
[3]*Grenoble Technical University, 51 Avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France*
[4]*Institut Universitaire de France, Paris, France*

SUMMARY

In this paper we consider the problem of scheduling on computing platforms composed of several independent organizations, known as the Multi-Organization Scheduling Problem (MOSP). Each organization provides both resources and jobs and follows its own objectives. We are interested in the best way to minimize the makespan on the entire platform when the organizations behave in a selfish way. We study the complexity of the MOSP problem with two different local objectives—makespan and average completion time—and show that MOSP is strongly NP-Hard in both cases. We formally define a selfishness notion, by means of restrictions on the schedules. We prove that selfish behavior imposes a lower bound of 2 on the approximation ratio for the global makespan. We present various approximation algorithms of ratio 2 which validate selfishness restrictions. These algorithms are experimentally evaluated through simulation, exhibiting good average performances and presenting good fairness to organizations' local objectives.

Received 15 November 2010; Revised 15 March 2011; Accepted 15 March 2011

KEY WORDS: scheduling; approximation algorithms; multi-organization scheduling

1. INTRODUCTION

1.1. Motivation and presentation of the problem

The new generation of many-core machines and the now mature grid computing systems allow the creation of unprecedented massively distributed systems. In order to fully exploit such large numbers of processors and cores available and reach the best performances, we need sophisticated scheduling algorithms that encourage users to share their resources and, at the same time, respect each user's own interests.

Many of these new computing systems are composed of organizations that own and manage clusters of computers. A user of such systems submits his/her jobs to a scheduling system that can choose any available machine in any of these clusters. However, each organization that shares its resources aims to take the maximum advantage of its own hardware. In order to improve cooperation between the organizations, local jobs should be prioritized to ensure that local performance objectives will be satisfied.

To find an efficient schedule for the jobs using the available machines is a crucial problem. Although each user submits jobs locally in his/her own organization and has its own performance

*Correspondence to: Denis Trystram, Grenoble Technical University, 51 Avenue Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France.
†E-mail: Denis.Trystram@imag.fr

goals, it is necessary to optimize the allocation of the jobs for the whole platform in order to achieve good performance.

This work extends the paper presented in the 2010 Euro-Par conference [1], by providing a new algorithm to solve the problem and deeper results on the quality of the performances obtained. We also study the notion of fairness on the results obtained by our algorithms.

1.2. Related work

From the classical scheduling theory, the problem of scheduling parallel jobs is related to Strip packing [2]. It corresponds to packing a set of rectangles (without rotations and overlaps) into a strip of machines in order to minimize the height used. Then, this problem was extended to the case where the rectangles were packed into a finite number of strips [3, 4]. More recently, an asymptotic $(1+\varepsilon)$-approximation AFPTAS with additive constant $\mathcal{O}(1)$ and with running-time polynomial in n and in $1/\varepsilon$ was presented in [5].

Schwiegelshohn *et al.* [6] studied a very similar problem, where the jobs can be scheduled in non-contiguous processors. Their algorithm is a 3-approximation for the maximum completion time (makespan) if all jobs are known in advance, and a 5-approximation for the makespan on the online, non-clairvoyant case.

The Multi-Organization Scheduling problem (MOSP) was introduced by Pascual *et al.* [7, 8] and studies how to efficiently schedule parallel jobs in new computing platforms, while respecting users' own selfish objectives. A preliminary analysis of the scheduling problem on homogeneous clusters was presented with the target of minimizing the makespan, resulting in a centralized 3-approximation algorithm. This problem was then extended for relaxed local objectives in [9].

The notion of cooperation between different organizations and the study of the impact of users' selfish objectives are directly related to Game Theory. The study of the Price of Anarchy [10] on non-cooperative games allows to analyze how far the social costs—results obtained by selfish decisions—are from the social optimum on different problems. In selfish load-balancing games (see [11] for more details), selfish agents aim to allocate their jobs on the machine with the smallest load. In these games, the social cost is usually defined as the completion time of the last job to finish (makespan). Several works studied this problem focusing on various aspects, such as convergence time to a Nash equilibrium [12], characterization of the worst-case equilibria [13], etc. We do not target here such game theoretical approaches.

1.3. Contributions and road map

As suggested in the previous section, the problem of scheduling in multi-organization clusters has been studied from several different view points. In this paper, we propose a theoretical analysis of the problem using classical combinatorial optimization approaches.

Our main contribution is the extension and analysis of the problem for the case in which sequential jobs are submitted by *selfish organizations* that can handle different local objectives (namely, makespan and average completion times).

We introduce new restrictions to the schedule that take into account the notion of *selfish organizations*, i.e. organizations that refuse to cooperate if their objectives could be improved just by executing earlier one of their jobs in one of their own machines. The formal description of the problem and the notations used in this paper are described in Section 2.

A fairness study is conducted using two different fairness metrics. The objective of this study is to evaluate if the results obtained by all organizations equally improve every organization's local objective by constructing schedules that are considered fair according to these metrics. Unfairness in the results obtained by each organization could be considered a reason to make an organization stop cooperation with the others.

Section 3 shows that any algorithm respecting our new selfishness restrictions cannot achieve approximation ratios better than 2 and that both problems are intractable. 2-approximation algorithms for solving the problem are presented in Section 4. Those heuristics are analyzed using fairness metrics presented in Section 5. Simulation experiments, discussed in Section 6, show

DOI: 10.1002/cpe

the good results obtained by our algorithms in average and how each organization perceives the fairness of the results obtained.

2. PROBLEM DESCRIPTION AND NOTATIONS

In this paper, we are interested in the scheduling problem in which different *organizations* own a physical cluster of identical machines that are interconnected. They share resources and exchange jobs with each other in order to simultaneously maximize the profits of the collectivity and their own interests. All organizations intent to minimize the global makespan (i.e. the maximum completion time in any local schedule) while they individually try to minimize their own objectives—either the makespan or the average completion time of their own jobs—in a selfish way.

Although each organization accepts to cooperate with others in order to minimize the global makespan, individually it behaves in a selfish way. An organization can refuse to cooperate if in the final schedule one of its migrated jobs could be executed earlier in one of the machines owned by the organization.

Formally, we define our target platform as a grid computing system with N different organizations interconnected by a middleware. Each organization $O^{(k)}$ $(1 \leq k \leq N)$ has $m^{(k)}$ identical machines available that can be used to run jobs submitted by users from any organization.

Each organization $O^{(k)}$ has $n^{(k)}$ jobs to execute, and the total number of jobs is denoted by $n = \sum_k n^{(k)}$. Each job $J_i^{(k)}$ $(1 \leq i \leq n^{(k)})$ will use one processor for exactly $p_i^{(k)}$ units of time[‡]. The size of the largest job from organization $O^{(k)}$ is denoted as $p_{\max}^{(k)}$. No preemption is allowed, i.e. after its activation, a job runs until its completion at time $C_i^{(k)}$.

We denote the makespan of a particular organization k by $C_{\max}^{(k)} = \max_{1 \leq i \leq n^{(k)}}(C_i^{(k)})$ and its sum of completion times as $\sum C_i^{(k)}$. The global makespan for the entire grid computing system is defined as $C_{\max} = \max_{1 \leq k \leq N}(C_{\max}^{(k)})$.

2.1. Local constraint

The *MOSP*, as first described in [7] consists in minimizing the global makespan ($C_{\max}$) with an additional *local constraint*: at the end, no organization can have its makespan increased if compared with the makespan that the organization could have by scheduling the jobs in its own machines ($C_{\max}^{(k)\text{local}}$). More formally, we call MOSP($C_{\max}$) the following optimization problem:

$$\text{minimize } C_{\max} \text{ such that for all } k(1 \leq k \leq N), \quad C_{\max}^{(k)} \leq C_{\max}^{(k)\text{local}}.$$

In this work, we also study the case where all organizations are locally interested in minimizing their average completion time while minimizing the global makespan. As in MOSP($C_{\max}$), each organization imposes that the sum of completion times of its jobs cannot be increased if compared with what the organization could have obtained using only its own machines ($\sum C_i^{(k)\text{local}}$). We denote this problem MOSP($\sum C_i$) and the goal of this optimization problem is to

$$\text{minimize } C_{\max} \text{ such that for all } k(1 \leq k \leq N), \quad \sum C^{(k)} \leq \sum C_i^{(k)\text{local}}.$$

2.2. Selfishness

In both MOSP($C_{\max}$) and MOSP($\sum C_i$), while the global schedule might be computed by a central entity, the organizations keep control on the way they execute the jobs in the end. This property means that, in theory, it is possible for organizations to cheat the devised global schedule by re-inserting their jobs earlier in the local schedules.

[‡]All machines are identical, i.e. every job will be executed at the same speed irrespective of the chosen machine.

DOI: 10.1002/cpe

In order to prevent such behavior, we define a new restriction on the schedule, called *selfishness restriction*. The idea is that, in any schedule respecting this restriction, no single organization can improve its local schedule by cheating.

Given a fixed schedule, let $J_f^{(l)}$ be the first foreign job scheduled to be executed in $O^{(k)}$ (or the first idle time if $O^{(k)}$ has no foreign job) and $J_i^{(k)}$ any job belonging to $O^{(k)}$. Then, the *selfishness restriction* forbids any schedule where $C_f^{(l)} - p_f^{(l)} < C_i^{(k)} - p_i^{(k)}$. In other words, $O^{(k)}$ refuses to cooperate if one of its jobs could be executed earlier in one of $O^{(k)}$ machines even if this leads to a larger global makespan.

3. COMPLEXITY ANALYSIS

3.1. Lower bounds

Pascual *et al.* [7] showed with an instance composed of two organizations and two machines per organization that every algorithm that solves MOSP (for rigid, parallel jobs and $C_{\max}$ as local objectives) has at least a $\frac{3}{2}$-approximation ratio when compared to the optimal makespan that could be obtained *without the local constraints*. We show that the same bound applies asymptotically even with a larger number of organizations.

Take the instance depicted in Figure 1(a). $O^{(1)}$ initially has two jobs of size N and all the others initially have N jobs of size 1. All organizations contribute only with one machine each. The optimal makespan for this instance is $N+1$ (Figure 1(b)), nevertheless, it delays jobs from $O^{(2)}$ and, as a consequence, does not respect MOSP's local constraints. The best possible makespan that respects the local constraints (whenever the local objective is the makespan or the average completion time) is $3N/2$, as shown in Figure 1(c).

3.2. Selfishness and lower bounds

Although all organizations will likely cooperate with each other to achieve the best global makespan possible, their selfish behavior will certainly impact the quality of the best attainable global makespan.

We study here the impact of new selfishness restrictions on the quality of the achievable schedules. We show that these restrictions impact MOSP($C_{\max}$) and MOSP($\sum C_i$) as compared with unrestricted schedules and, moreover, that MOSP($C_{\max}$) with selfishness restrictions suffers from limited performances compared to MOSP($C_{\max}$) with local constraints.

Proposition 1
No approximation algorithm for both MOSP($C_{\max}$) and MOSP($\sum C_i$) has a ratio asymptotically better than 2 regarding the optimal makespan *without constraints* if all organizations behave selfishly.

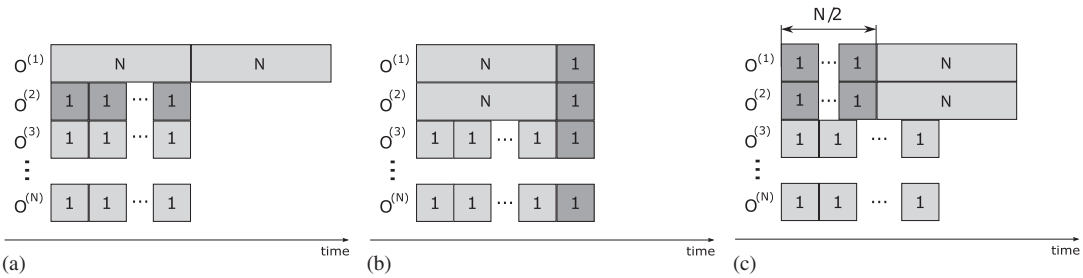

Figure 1. Ratio between global optimum makespan and the optimum makespan that can be obtained for both MOSP($C_{\max}$) and MOSP($\sum C_i$). Jobs owned by organization $O^{(2)}$ are highlighted. (a) Initial instance—$C_{\max} = 2N$. (b) Global optimum without constraints—$C_{\max} = N+1$. (c) Optimum with MOSP constraints—$C_{\max} = 3N/2$.

DOI: 10.1002/cpe

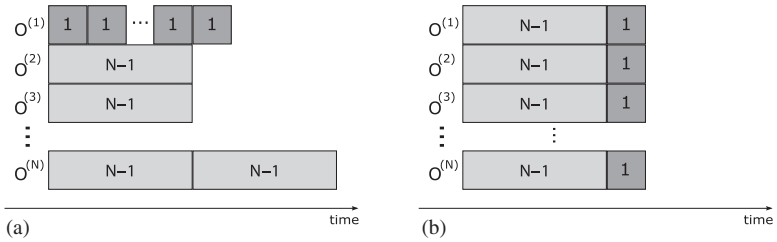

Figure 2. Ratio between global optimum makespan with MOSP constraints and the makespan that can be obtained by MOSP($C_{\max}$) with selfish organizations: (a) initial instance—$C_{\max} = 2N-2$ and (b) global optimum with MOSP constraints—$C_{\max} = N$.

Proof
We prove this result by using the example described in Figure 1. It is clear from Figure 1(b) that an optimal solution for a schedule without local constraints can be achieved in $N+1$. However, with added selfishness restrictions, Figure 1(a) (with a makespan of $2N$) represents the only possible valid schedule. We can, therefore, conclude that local constraints combined with selfishness restrictions imply that no algorithm can provide an approximation ratio asymptotically better than 2 when compared with the problem without constraints. □

Proposition 1 gives a ratio regarding the optimal makespan without the local constraints imposed by MOSP. We can show that the same approximation ratio of 2 also applies for MOSP($C_{\max}$) regarding the optimal makespan even if MOSP constraints are respected.

Proposition 2
Any approximation algorithm for MOSP($C_{\max}$) has a ratio greater than or equal to $2-(2/N)$ regarding the optimal makespan *with local constraints* if all organizations behave selfishly.

Proof
Take the instance depicted in Figure 2(a). $O^{(1)}$ initially has N jobs of size 1 and $O^{(N)}$ has two jobs of size $N-1$. The optimal solution that respects MOSP local constraints is given in Figure 2(b) and has $C_{\max}$ equal to N. Nevertheless, the best solution that respects the selfishness restrictions is the initial instance with a $C_{\max}$ equal to $2N-2$. Thus, the ratio of the optimal solution with the selfishness restrictions to the optimal solution with MOSP constraints is $2-(2/N)$. □

3.3. Computational complexity

This section studies how difficult it is to find optimal solutions for the MOSP problem. We consider the decision version of the MOSP defined as follows:

Instance: A set of N organizations (for $1 \leq k \leq N$, organization $O^{(k)}$ has $n^{(k)}$ jobs, $m^{(k)}$ identical machines, and makespan as the local objective) and an integer ℓ.

Question: Does there exist a schedule with a makespan less than ℓ that respects the local constraints?

Theorem 1
MOSP($C_{\max}$) and MOSP($\sum C_i$) are strongly NP-complete.

A reduction from the classical 3-PARTITION problem [14] (see definition below) to MOSP is straightforward and it is schematized in Figure 3. This reduction, however, represents a degenerate case of the MOSP problem. The instance presents a configuration where initially one organization is very loaded and some organizations do not contribute with any work. We believe that this configuration misrepresents the concept of cooperation introduced by the MOSP problem.

We present in this section a slightly more complex proof for the complexity of MOSP($C_{\max}$) and MOSP($\sum C_i$) with an instance where any organization could potentially have its makespan

DOI: 10.1002/cpe

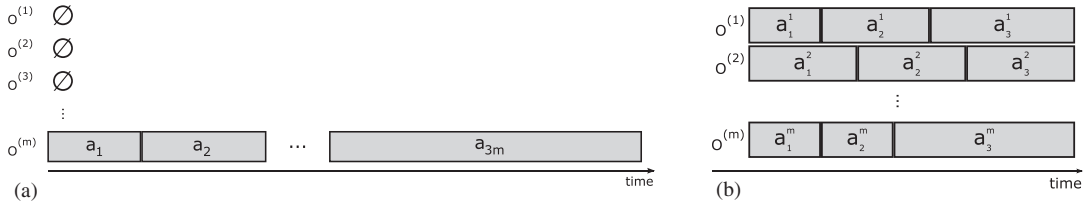

Figure 3. Direct reduction of MOSP from 3-PARTITION: (a) initial instance and (b) optimum.

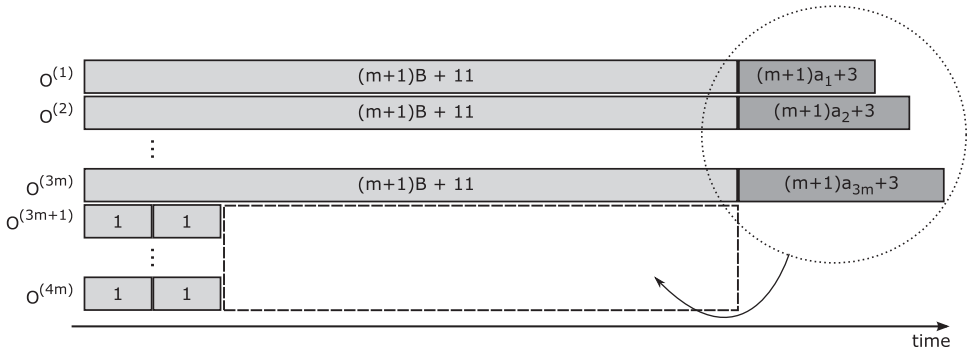

Figure 4. Reduction of MOSP($C_{\max}$) and MOSP($\sum C_i$) from 3-PARTITION.

improved by cooperating with others. We show that the problem remains difficult even for instances limited to two jobs per organization.

Theorem 2
MOSP($C_{\max}$) and MOSP($\sum C_i$) are strongly NP-complete even if each organization has exactly two jobs per organization.

Proof
It is straightforward to see that MOSP($C_{\max}$)$\in NP$ and MOSP($\sum C_i$)$\in NP$. Our proof is based on a reduction from the well-known 3-PARTITION problem [14]:

Instance: A bound $B \in Z^+$ and a finite set A of $3m$ integers $\{a_1, \ldots, a_{3m}\}$, such that every element of A is strictly between $B/4$ and $B/2$ and such that $\sum_{i=1}^{3m} a_i = mB$.

Question: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for all $1 \leq i \leq m$, $\sum_{a \in A_i} a = B$ and A_i is composed of exactly three elements?

Given an instance of 3-PARTITION, we construct an instance of MOSP where, for $1 \leq k \leq 3m$, organization $O^{(k)}$ initially has two jobs $J_1^{(k)}$ and $J_2^{(k)}$ with $p_1^{(k)} = (m+1)B + 11$ and $p_2^{(k)} = (m+1)a_k + 3$, and all other organizations have two jobs with processing time equal to 1. We then set ℓ to be equal to $(m+1)B + 11$. Figure 4 depicts the described instance. This construction is performed in polynomial time. Now, we prove that A can be split into m disjoint subsets $A_1, \ldots, A_m$, each one summing up to B, if and only if this instance of MOSP has a solution with $C_{\max} \leq (m+1)B + 11$.

Assume that $A = \{a_1, \ldots, a_{3m}\}$ can be partitioned into m disjoint subsets $A_1, \ldots, A_m$, each one summing up to B. In this case, we can build an optimal schedule for the instance as follows:

- for $1 \leq k \leq 3m$, $J_1^{(k)}$ is scheduled on machine k;
- for $3m+1 \leq k \leq 4m$, $J_1^{(k)}$ and $J_2^{(k)}$ are scheduled on machine k;
- for $1 \leq i \leq m$, let $A_i = \{a_{i_1}, a_{i_2}, a_{i_3}\} \subseteq A$. The jobs $J_2^{(a_{i_1})}$, $J_2^{(a_{i_2})}$, and $J_2^{(a_{i_3})}$ are scheduled on machine $3m+i$.

This construction provides a schedule where the global $C_{\max}$ is the optimal one. Also, it is easy to see that the local constraints for MOSP($C_{\max}$) and MOSP($\sum C_i$) are respected: for MOSP($C_{\max}$),

DOI: 10.1002/cpe

it is sufficient to note that no organization has its makespan increased; for MOSP($\sum C_i$), note that each job either remains in its original organization and schedule (lighter colored jobs in Figure 4) or is migrated to a different organization at an earlier time (darker colored jobs in Figure 4), strictly decreasing the average completion time.

Conversely, assume that MOSP has a solution with $C_{\max} \leq (m+1)B+11$. The total work (W) of the jobs that must be executed is $W = 3m \cdot ((m+1)B+11) + \sum_{i=1}^{3m}((m+1)B+a_i) + m \cdot 2 = 4m \cdot ((m+1)B+11)$. Since we have exactly $4m$ organizations, the solution must be the optimal solution and there are no idle times in the scheduling. Moreover, $3m$ machines must execute only one job of size $(m+1)B+11$. W.l.o.g, we can consider that for $3m+1 \leq k \leq 4m$, machine k performs jobs of size less than $(m+1)B+11$.

To prove our proposition, we first show two lemmas that characterize the structure of the solution for the MOSP problem:

Lemma 1
For all $3m+1 \leq k \leq 4m$, exactly three jobs of size not equal to 1 must be scheduled on machine k if $C_{\max}^{(k)} = (m+1)B+11$.

Proof
We prove this lemma by contradiction. First, note that for all integers from the 3-PARTITION instance, it holds that $a_i > B/4$.

Assume that one machine has less than three jobs of size not equal to 1. $3m$ jobs of this kind must be assigned to the last m organizations (since the first $3m$ organizations executes each one large job of size $(m+1)B+11$). If one machine has less than three jobs, then we must have at least one other machine with four or more jobs of this kind assigned to it. The makespan on this other machine must be of at least $4 \cdot ((m+1)a_i+3) > 4 \cdot ((m+1)B/4+3) = (m+1)B+12$. This contradicts the fact that $C_{\max}^{(k)} = (m+1)B+11$ and shows that exactly three jobs of size not equal to 1 must be scheduled on each machine. □

Lemma 2
For all $3m+1 \leq k \leq 4m$, exactly two jobs of size 1 are scheduled on each machine if $C_{\max}^{(k)} = (m+1)B+11$.

Proof
Since $C_{\max}^{(k)} = (m+1)B+11$, each large job of size $(m+1)B+11$ must remain in its original organization. Consequently, the $2m$ jobs of size 1 must be scheduled in one of the m organizations not having a large job assigned to it. However, each of these organizations have $C_{\max}^{(k)\text{local}} = 2$, which means that any migration that results in a schedule with one organization not having two jobs of size 1 will result in a schedule that does not respect both MOSP($C_{\max}$) and MOSP($\sum C_i$) local constraints. □

From the solution for the MOSP problem, we construct m disjoint subsets $A_1, A_2, \ldots, A_m$ of A as follows: for all $1 \leq i \leq m$, a_j is in A_i if the job with size $(m+1)a_j+3$ is scheduled on machine $3m+i$. Note that all elements of A belong to one and only one set in $\{A_1, \ldots, A_m\}$. We prove that A is a partition with the desired properties.

We focus on organization $O^{(3m+i)}$, where A_i was assigned. Lemmas 1 and 2 show how the jobs will be assigned to this organization. First, Lemma 1 shows that A_i is composed of exactly three jobs that will be executed on $O^{(3m+i)}$. Second, Lemma 2 shows that exactly two jobs of size 1 will be scheduled at the beginning of the schedule.

Since all organizations must have a local $C_{\max}$ exactly equal to $(m+1)B+11$, we have that $C_{\max}^{(3m+i)} = (m+1)B+11 = 1+1+\sum_{a_j \in A_i}((m+1)a_j+3)$, which implies that:

$$2+\sum_{a_j \in A_i}((m+1)a_j+3) = (m+1)B+11 \Rightarrow \sum_{a_j \in A_i}(m+1)a_j = (m+1)B \Rightarrow \sum_{a_j \in A_i} a_j = B.$$

DOI: 10.1002/cpe

Thus, we can deduce that A_i is composed of exactly three elements and $\sum_{a_j \in A_i} a_j = B$. In other words, $\{A_i, \ldots, A_m\}$ is a solution for the 3-PARTITION problem. □

4. ALGORITHMS

In this section, we present four different heuristics to solve MOSP($C_{\max}$) and MOSP($\sum C_i$). All algorithms present the additional property of respecting selfishness restrictions.

4.1. Iterative load balancing algorithm (ILBA)

ILBA [8] is a heuristic that redistributes the load from the most loaded organizations.

The idea is to incrementally rebalance the load without delaying any job. First, the less loaded organizations are rebalanced. Then, one-by-one, each organization has its load rebalanced.

The heuristic works as follows. First, each organization schedules its own jobs locally and the organizations are enumerated by non-decreasing makespans, i.e. $C_{\max}^{(1)} \leq C_{\max}^{(2)} \leq \cdots \leq C_{\max}^{(N)}$. For $k=2$ until N, jobs from $O^{(k)}$ are rescheduled sequentially, and assigned to the less loaded of organizations $O^{(1)} \ldots O^{(k)}$.

Each job is rescheduled by ILBA either earlier or at the same time that the job was scheduled before the migration. In other words, no job is delayed by ILBA, which guarantees that the local constraint is respected for MOSP($C_{\max}$) and MOSP($\sum C_i$).

ILBA has the same time complexity as any list scheduling algorithm for the classical problem of minimizing the makespan. Its time complexity is $\mathcal{O}(n \log n)$. We recall that n is the total number of jobs.

4.2. LPT-LPT and SPT-LPT heuristics

We developed and evaluated (see Section 6) two new heuristics based on the classical LPT (Longest Processing Time First [15]) and SPT (Smallest Processing Time First [16]) algorithms for solving MOSP($C_{\max}$) and MOSP($\sum C_i$), respectively. Both heuristics work in two phases. During the first phase, all organizations minimize their own local objectives. Each organization starts applying LPT for its own jobs if the organization is interested in minimizing its own makespan, or starts applying SPT if the organization is interested in its own average completion time.

The second phase is when all organizations cooperatively minimize the makespan of the entire grid computing system without worsening any local objective. This phase works as follows: each time an organization becomes idle, i.e. it finishes the execution of all jobs assigned to it, the longest job that has not started yet is migrated and executed by the idle organization. This greedy algorithm works like a global LPT, always choosing the longest job yet to be executed among jobs from all organizations.

The first and second phases correspond to a list scheduling algorithm, hence the total running time of both algorithms is $\mathcal{O}(n \log n)$.

4.3. Less Helped First heuristic

Like the previous algorithms, Less Helped First works in two phases. First, each organization $O^{(k)}$ is classified according to the total work originally submitted by a user in $O^{(k)}$ that was actually executed by a different organization. Locally, the order of the jobs is arbitrary (it could be the submission order, for instance).

Second, each time a processor becomes idle, the algorithm will prioritize the organization that received *less help* from the others, i.e. that had less work executed by other organizations. The algorithm will migrate any job from the *less helped* organization that has not started yet to be executed by the idle processor.

The running time of the first phase is $\mathcal{O}(n \log n)$. During the second phase, after each job migration, the classification of the organizations must be updated. This update can be done in

DOI: 10.1002/cpe

$\mathcal{O}(\log n)$ and there are at most n migrations (one per job). Since the number of organizations is negligible compared to the number of jobs, the total running time of the algorithm is $\mathcal{O}(n \log n)$.

4.4. Analysis

ILBA, LPT-LPT, SPT-LPT, and Less Helped First do not delay any of the jobs when compared to the initial local schedule. During the rebalancing phase, all jobs either remain in their original organization or are migrated to an organization that became idle at a preceding time. The implications are

- the *selfishness* restriction is respected—if a job is migrated, it will start before any foreign jobs that may have been assigned to its original organization;
- if organizations' local objective is to minimize the makespan, migrating a job to a previous moment in time will decrease the job's completion time and, as a consequence, it will not increase the initial makespan of the organization;
- if organizations' local objective is to minimize the average completion time, migrating a job from the initial organization to another that became idle at a previous moment in time will not increase the completion time of any job. This means that the $\sum C_i$ of the jobs from the initial organization is either decreased or remains constant;
- the rebalancing phase of all four algorithms works as the list scheduling algorithms. Graham's classical approximation ratio $2-(1/N)$ of list scheduling algorithms [15] holds for all of them.

Proposition 1 (Section 3.2) states that no algorithm respecting selfishness restrictions can achieve an approximation ratio for MOSP($C_{\max}$) better than 2 regarding the optimal makespan without constraints. Since all our algorithms reach an approximation ratio of 2, no further enhancements are possible without removing selfishness restrictions.

5. FAIRNESS

Another form of encouragement of cooperation between the organizations is providing algorithms that equally improve their local objectives. Organizations that are not invidious of other organizations are more likely to share their resources to the platform. Selfish organizations could potentially become invidious if the results obtained when solving the MOSP problem improve the local objectives of the organizations in an unfair manner.

Some fairness metrics can be found in the literature—Variance, Coefficient of Variance (Variance/Mean), Min-Max ratio, etc. The Jain Index [17] is nowadays one of the most popular metrics used to measure fairness [18]. We also studied the *Stretch* as an index of fairness, as we will show in the subsequent sections.

In this section we study the fairness of the results produced by the algorithms presented in Section 4. Using the same workload generated for the experimental results shown in Section 6, we evaluate the fairness of these algorithms for two important metrics: Stretch and the Jain Index.

5.1. Stretch

Stretch is a performance metric that was extensively studied in the scheduling literature. This performance metric can also be considered as a fairness metric. We adapted the notion of Stretch to the MOSP problem, to measure the Stretch of an entire organization. We define Stretch as the ratio of the local $C_{\max}$ obtained by the organization to the $C_{\max}$ that the organization could have obtained if it had all the machines available only for its jobs (denoted by $C_{\max}^{(k)\text{alone}}$). Formally, the Stretch is calculated as follows:

$$\text{Stretch} = \max_k \frac{C_{\max}^{(k)}}{C_{\max}^{(k)\text{alone}}}.$$

DOI: 10.1002/cpe

Since $C_{\max}^{(k)\text{alone}}$ is bounded by the theoretical lower bound $\sum_i p_i^{(k)}/N = C_{\max}^{(k)\text{local}}/N$ and since MOSP($C_{\max}$) constraint guarantees that $C_{\max}^{(k)} \leq C_{\max}^{(k)\text{local}}$, we have

$$\text{Stretch} = \max_k \frac{C_{\max}^{(k)}}{C_{\max}^{(k)\text{alone}}} \leq \frac{C_{\max}^{(k)\text{local}}}{\frac{C_{\max}^{(k)\text{local}}}{N}} \leq N.$$

Note that this upper bound for the Stretch metric is tight. Consider the instance with N organizations, each one having N jobs with processing times equal to 1. $C_{\max}^{(k)\text{alone}}$ is equal to 1 for all organizations k, and $C_{\max}^{(k)\text{local}} = N$. There is only one scheduling that respects MOSP($C_{\max}$) local constraint, the one that schedules all jobs in their original organizations.

5.2. Jain Index

Originally, the Jain Index was introduced for the study of resource sharing and allocation problems. The Jain Index of a set of n users receiving an 'allocation' x_i [17] is defined as

$$f(x) = \frac{\left(\sum_{i=1}^n x_i\right)^2}{n \sum_{i=1}^n x_i^2}, \quad x_i \geq 0. \tag{1}$$

If the values of x_i follow a certain random distribution, the index is the ratio of the first moment (the mean) to the second moment of the distribution.

The Jain Index has the following useful properties:

- it is independent of the size of the population;
- it is independent of the scale (or metric unit) used;
- the values of the index are bound between $1/n$ and 1;
- it is continuous.

We evaluated the fairness of the heuristics presented in Section 4 regarding the improvements obtained on the local $C_{\max}$ by each organization. We define this *improvement* as the ratio of the obtained $C_{\max}$ to the theoretical lower bound of each organization over all available machines of the system.

The lower bound of each organization is calculated by $LB^{(k)} = \max(\sum_i \frac{p_i^{(k)}}{N}, p_{\max}^{(k)})$. The improvement for organization $O^{(k)}$ (the x_k of Equation 1) is then defined as: $C_{\max}^{(k)}/LB^{(k)}$. Thus, the final Jain Index calculated to measure the fairness of the improvements can be stated as follows:

$$\text{Jain Index} = \frac{\left(\sum_{k=1}^N \frac{C_{\max}^{(k)}}{LB^{(k)}}\right)^2}{N \sum_{k=1}^N \left(\frac{C_{\max}^{(k)}}{LB^{(k)}}\right)^2}.$$

6. EXPERIMENTS

We conducted an extensive series of simulations comparing ILBA, LPT-LPT, SPT-LPT, and Less Helped First under various experimental settings. The workload was randomly generated with parameters matching the typical environment found in academic grid computing systems [8].

We generated random workloads with $N = 5$, 10, and 20 organizations. For each number of organizations, we have generated instances with different number of total jobs (from 100 to 1000), with sizes chosen uniformly at random from 1 to 1000. For each combination of these parameters, we generated 20 000 random instances. In our tests, the number of initial jobs in each organization

DOI: 10.1002/cpe

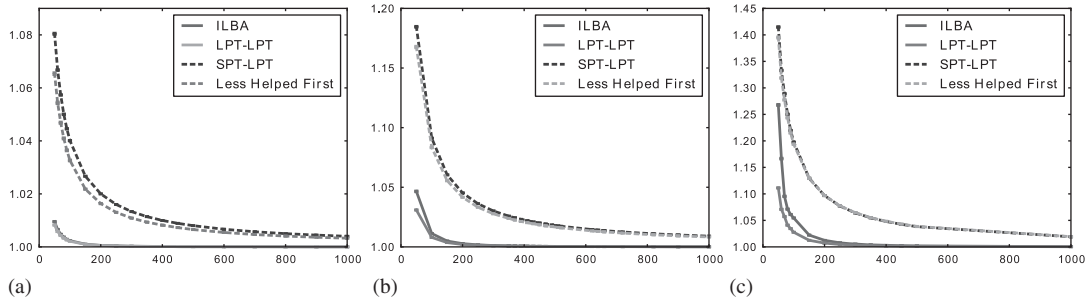

Figure 5. Mean and confidence interval of the measured $C_{\max}^{(k)}$ obtained by ILBA, LPT-LPT, SPT-LPT, and Less Helped First: (a) $N=5$; (b) $N=10$; and (c) $N=20$.

follows a Zipf distribution with exponent equal to 1.4267, which best models virtual organizations in real-world grid computing systems [19]. All results are presented with confidence level of 95%.

6.1. Global $C_{\max}$ analysis

We are interested in the improvement of the global $C_{\max}$ provided by the different algorithms. The results are evaluated with a comparison to $C_{\max}$ obtained by the algorithms with the well-known theoretical lower bound for the scheduling problem without constraints $LB = \max(\sum_{i,k} \frac{p_i^{(k)}}{m^{(k)}}, p_{\max})$.

Our main conclusion is that, despite the fact that the selfishness restrictions are respected by all heuristics, ILBA and LPT-LPT obtained near optimal results for most cases. This is not unusual, since it follows the patterns of experimental behavior of standard list scheduling algorithms, in which it is easy to obtain a near optimal schedule when the number of jobs grows large. SPT-LPT and Less Helped First produce worse results due to the effect of the local order of the jobs. Figure 5 shows the average global $C_{\max}$ obtained by the heuristics for instances with different number of organizations and the total number of jobs.

However, in some particular cases, in which the number of jobs is not much larger than the number of machines available, the experiments yield more interesting results. Figure 6 shows the histogram of a representative instance of such a particular case. The histograms show the frequency of the ratio $C_{\max}$ obtained to the lower bound over 20 000 different instances with 20 organizations and 100 jobs for ILBA, LPT-LPT, SPT-LPT, and Less Helped First. Similar results have been obtained for many different sets of parameters. LPT-LPT outperforms ILBA (and SPT-LPT) for most instances and its average ratio to the lower bound is less than 1.3. Less Helped First is the heuristic that produces the worst results.

6.2. Local $C_{\max}$ analysis

We are also interested in the improvements obtained by each organization that our four algorithms provide. In order to have a global view of the total improvement obtained, we have evaluated the sum of the local makespans obtained by all organizations: $\sum_k C_{\max}^{(k)}$. The sum of the local makespans is important because it shows the average results that each selfish organization obtains.

The experiments show that ILBA produces results with a lower makespan in average. LPT-LPT, SPT-LPT, and Less Helped First produce similar results in average. Recall from Section 4.1 that ILBA rebalances the load of all organizations, from the less loaded to the more loaded organization. This mechanism minimizes the makespan of the organization individually in a more aggressive way and, therefore, produces lower average local makespans. Figure 7 shows the average local $C_{\max}$ obtained by all heuristics.

It is interesting to note that the gap between the average result obtained by ILBA and the other heuristics increases as the number of organizations and the total number of jobs increases.

DOI: 10.1002/cpe

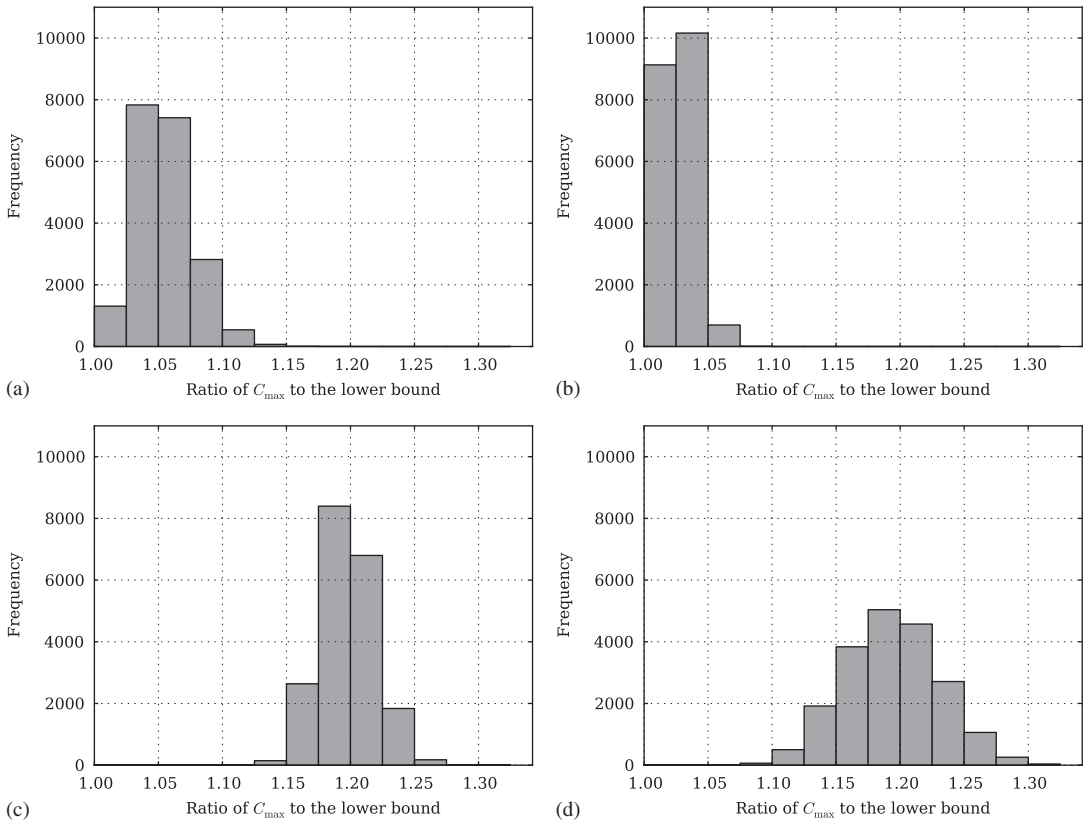

Figure 6. Frequency of results obtained by ILBA, LPT-LPT, SPT-LPT, and Less Helped First for $N = 20$ and $n = 100$: (a) ILBA; (b) LPT-LPT; (c) SPT-LPT; and (d) Less Helped First.

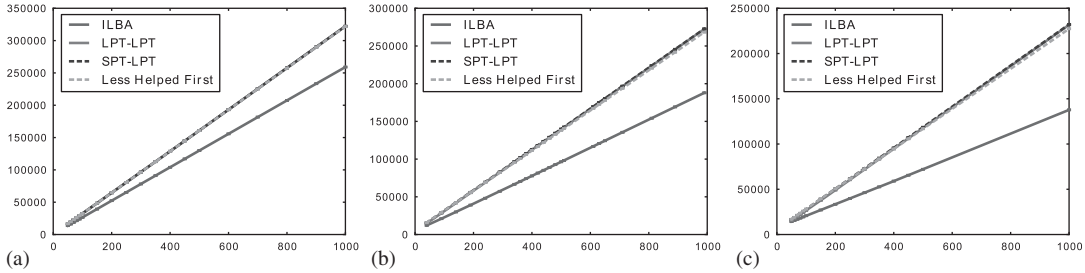

Figure 7. Mean and confidence interval of the measured $\sum_k C_{\max}^{(k)}$ obtained by ILBA, LPT-LPT, SPT-LPT, and Less Helped First: (a) $N = 5$; (b) $N = 10$; and (c) $N = 20$.

6.3. Fairness analysis

In the two previous sections we have analyzed the performance obtained by the four algorithms presented in Section 4 for the global makespan obtained and for the local makespan obtained by each organization. In this section we use the same testbed to study how each organization perceives the fairness of these results. To study the fairness, we use the Stretch and Jain Index metrics presented in Section 5.

6.3.1. Stretch. We have measured the worst—i.e. maximum—Stretch (as defined in Section 5.1) obtained during the simulations. Figure 8 shows how the Stretch varies according to the total number of jobs of the system.

DOI: 10.1002/cpe

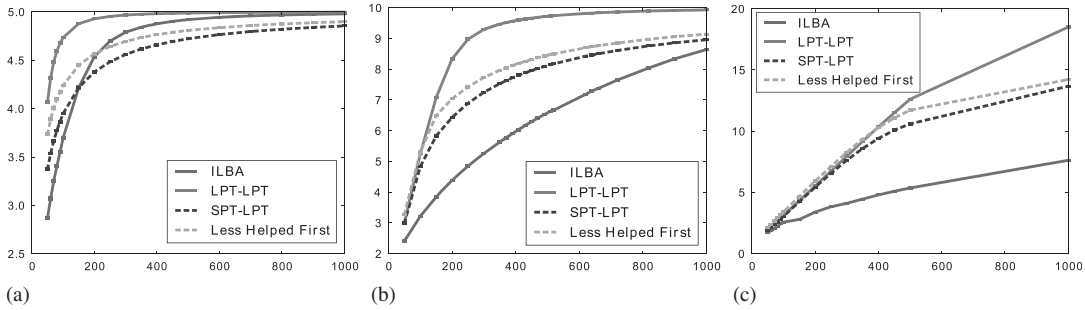

Figure 8. Mean and confidence interval of the measured *Stretch* obtained by ILBA, LPT-LPT, SPT-LPT, and Less Helped First: (a) $N=5$; (b) $N=10$; and (c) $N=20$.

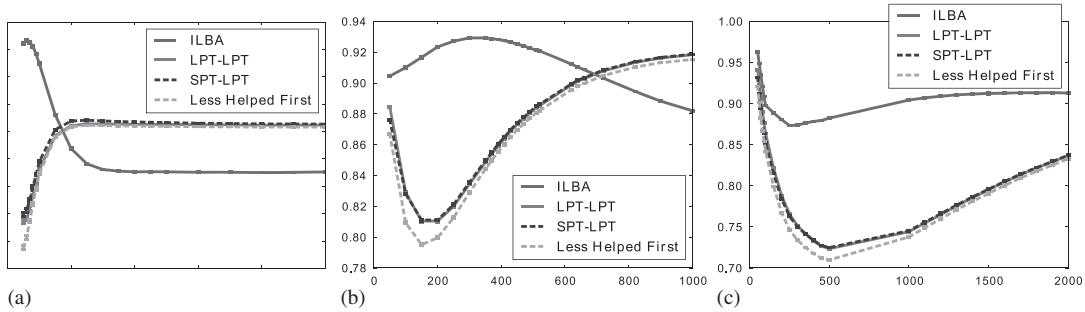

Figure 9. Mean and confidence interval of the measured *Jain Index* obtained by ILBA, LPT-LPT, SPT-LPT, and Less Helped First: (a) $N=5$; (b) $N=10$; and (c) $N=20$.

The results show that the Stretch increases asymptotically to the upper bound of the metric (i.e. the number of organizations) as the total number of jobs on the system increases.

The Stretch obtained with our algorithms is given by the less loaded organization. Our heuristics never migrate jobs from the organization that has the lower local $C_{\max}$. For this organization, the $C_{\max}^{(k)}$ will always be equal to the $C_{\max}^{(k)\text{local}}$ and the Stretch will be higher. The Zipf distribution used to randomly assign the jobs to the organizations distributes more jobs to the less loaded organization when the total number of organizations is smaller. For this reason, the average Stretch grows faster for $N=5$.

When the number of jobs is small, the performance of ILBA degrades more slowly because ILBA rebalances first the less loaded organizations, while the others rebalance the organizations in a dynamic order.

LPT-LPT presented the worst values for Stretch. Bigger Stretch values indicate that organizations are more penalized by the jobs from other organizations. Recall from Section 6 that LPT-LPT produces better results for the global $C_{\max}$. To achieve these results, LPT-LPT must penalize some organizations, and Stretch measures exactly this behavior.

6.3.2. Jain Index. Jain Index and Stretch metrics measure two different forms of fairness. The former measures how each organization perceives the improvements on its own $C_{\max}$, while the latter shows how $C_{\max}$ of each individual organization is affected by the jobs from other organizations.

When the number of organizations N is 5, our experiments showed that the four heuristics provide fair results, with Jain Index higher than 0.88. Recall that the Jain Index is bounded from $1/N$ (unfair) to 1 (perfect fairness).

Figure 9(a) indicates that the ILBA heuristics have two moments. In the first, when the number of jobs is smaller, ILBA heuristic produces more fair results than the other heuristics. Its Jain Index varies from 0.92 up to 0.95.

DOI: 10.1002/cpe

During the second moment, when the number of jobs is higher, ILBA produces the less fair results between all the four heuristics. LPT-LPT, SPT-LPT, and Less Helped First produce results with similar fairness indices, while ILBA fairness is clearly worse when compared to the others. Recall that the index is almost constant when $N=5$ and there are more than 300 jobs on the platform.

The experiments with $N=10$ shows that ILBA produces results with fairness similar to the results with $N=5$. Its Jain Index is of about 0.90. When the number of jobs is over 700, LPT-LPT, SPT-LPT, and Less Helped First are fairer. Their Jain Index varies between 0.79 and 0.91.

For $N=20$, the difference between the performance of ILBA and the others are higher. ILBA has Jain Index of about 0.90 while the fairness of the other three heuristics decreases from 0.94 to 0.70. The number of jobs of the simulations shown in Figure 9(c) was increased up to 2000 to show that for $N=20$ the results indicate that ILBA has also two moments, like for the cases where $N=5$ and $N=10$.

Even if ILBA provided fairer results than the others with less jobs on the system, all four algorithms improve the local $C_{\max}$ obtained by each organization in a fair way.

7. CONCLUDING REMARKS

In this paper, we have investigated the scheduling on multi-organization platforms. We presented the MOSP($C_{\max}$) problem from the literature and extended it to a new related problem MOSP($\sum C_i$) with another local objective. In each case we studied how to improve the global makespan while guaranteeing that no organization will worsen its own results.

We showed first that both versions MOSP($C_{\max}$) and MOSP($\sum C_i$) of the problem are strongly NP-hard. Furthermore, we introduced the concept of *selfishness* in these problems which corresponds to additional scheduling restrictions designed to reduce the incentive for the organizations to cheat locally and disrupt the global schedule. We proved that any algorithm respecting selfishness restrictions cannot achieve a better approximation ratio than 2 for MOSP($C_{\max}$) regarding the optimal makespan without MOSP local constraints.

Three new scheduling algorithms were proposed, namely LPT-LPT, SPT-LPT, and Less Helped First in addition to ILBA from the literature. All these algorithms are list scheduling, and thus achieve a 2-approximation. We provided an in-depth analysis of these algorithms, showing that all of them respect the selfishness restrictions.

Finally, all these algorithms were implemented and analyzed through experimental simulations. The results show that when considering the global makespan, our new LPT-LPT outperforms ILBA, and that all algorithms exhibit near optimal global performances when the number of jobs becomes large.

Considering the local objectives, ILBA achieves better average local makespans. ILBA also proved to present more fair results according to the Stretch metric. All the four algorithms presented good fairness using the Jain Index, showing that the local makespan of all organizations is equally improved.

The future research directions will be more focused on game theory. We intend to study schedules in the case where several organizations secretly cooperate to cheat the central authority.

ACKNOWLEDGEMENTS

This work was partially sponsored by a Google Research Award and partially supported by the bilateral Brazilian-French CAPES-COFECUB program (Ma 660/10).

REFERENCES

1. Cohen J, Cordeiro D, Trystram D, Wagner F. Analysis of multi-organization scheduling algorithms. *Euro-Par 2010—Parallel Processing* (*Lecture Notes in Computer Science*, vol. 6272), D'Ambra P, Guarracino M, Talia D (eds.). Springer: Berlin/Heidelberg, 2010; 367–379.

DOI: 10.1002/cpe

2. Baker BS, Coffman EG Jr, Rivest RL. Orthogonal packings in two dimensions. *SIAM Journal on Computing* 1980; **9**(4):846–855.
3. Zhuk SN. Approximate algorithms to pack rectangles into several strips. *Discrete Mathematics and Applications* 2006; **16**(1):73–85.
4. Ye D, Han X, Zhang G. On-line multiple-strip packing. *Proceedings of the Third International Conference on Combinatorial Optimization and Applications* (*Lecture Notes in Computer Science*, vol. 5573), Berlin S. (ed.). Springer: Berlin, 2009; 155–165.
5. Jansen K, Otte C. Approximation algorithms for multiple strip packing. *Proceedings of Seventh Workshop on Approximation and Online Algorithms* (*WAOA*), Copenhagen, Denmark, 2009.
6. Schwiegelshohn U, Tchernykh A, Yahyapour R. Online scheduling in grids. *IEEE International Symposium on Parallel and Distributed Processing* (*IPDPS*), Miami, FL, U.S.A., 2008; 1–10.
7. Pascual F, Rzadca K, Trystram D. Cooperation in multi-organization scheduling. *Euro-Par 2007 Parallel Processing* (*Lecture Notes in Computer Science*, vol. 4641). Springer: Berlin, 2007; 224–233. DOI: http://dx.doi.org/10.1007/978-3-540-74466-5_25.
8. Pascual F, Rzadca K, Trystram D. Cooperation in multi-organization scheduling. *Concurrency and Computation*: *Practice & Experience* 2009; **21**(7):905–921. DOI: http://dx.doi.org/10.1002/cpe.v21:7.
9. Ooshita F, Izumi T. A generalized multi-organization scheduling on unrelated parallel machines. *International Conference on Parallel and Distributed Computing*, *Applications and Technologies* (*PDCAT*). IEEE Computer Society: Los Alamitos, CA, U.S.A., 2009; 26–33.
10. Koutsoupias E, Papadimitriou C. Worst-case equilibria. *Proceedings of 16th Annual Symposium on Theoretical Aspects of Computer Science* (*Lecture Notes in Computer Science*, vol. 1563). Springer: Berlin, Trier, Germany, 1999; 404–413.
11. Nisam N, Roughgarden T, Tardos E, Vazirani VV. *Algorithmic Game Theory*. Cambridge University Press: Cambridge, 2007.
12. Even-Dar E, Kesselman A, Mansour Y. Convergence time to nash equilibria. *ACM Transactions on Algorithms* 2007; **3**(3):32.
13. Caragiannis I, Flammini M, Kaklamanis C, Kanellopoulos P, Moscardelli L. Tight bounds for selfish and greedy load balancing. *Proceedings of the 33rd International Colloquium on Automata*, *Languages and Programming* (*Lecture Notes in Computer Science*, vol. 4051). Springer: Berlin, 2006; 311–322.
14. Garey MR, Johnson DS. *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*. W. H. Freeman: New York, 1979.
15. Graham RL. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* 1969; **17**(2):416–429.
16. Bruno JL, Coffman EG Jr, Sethi R. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM* 1974; **17**(7):382–387. DOI: http://doi.acm.org/10.1145/361011.361064.
17. Jain RK, Chiu DMW, Hawe WR. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report*, Digital Equipment Corporation, September 1984.
18. Legrand A, Touati C. How to measure efficiency?. *Proceedings of the First International Workshop on Game theory for Communication networks* (*Game-Comm'07*), Nantes, France, 2007.
19. Iosup A, Dumitrescu C, Epema D, Li H, Wolters L. How are real grids used? The analysis of four grid traces and its implications. *Seventh IEEE/ACM International Conference on Grid Computing*, Barcelona, Spain, 2006; 262–269.

DOI: 10.1002/cpe